

Compile 2013 VS Community Ov4.2-04.8.8



The officially supported Windows build platform for OpenCPN 4.1 beta to OpenCPN 4.8.8 is **Microsoft Visual Studio 2013** Express for Windows Desktop.

- Opencpn v4.2 to v4.8.8
- Microsoft Visual Studio 2013
- wxWidgets 3.0.2
- Plugin API started with API 1.13 and supports through API 1.15

It is still possible to perform the build with older versions of Visual Studio - particularly if you are still running Windows XP on your development machine, you cannot install a Visual Studio version newer than 2010. If what you have to do differs from the default VS2013 workflow, it is clearly noted in the instructions below.

Note: Older versions of OpenCPN, up to 4.0, used the VS2010 toolchain and wxWidgets 2.8.

Important note before you start

The order of the steps described below really matters. Don't skip any steps not explicitly marked as optional and don't change their order unless you really know what you are doing.

It's an excellent idea to read the whole text first and make sure you understand what it's talking about, especially if you are new to software development.

If you encounter any problems, please get to us in the forum and tell us where you are failing so we can help you and improve these instructions for the others.

Before you go to the support forum and start to complain how much these instructions don't work, try to really follow them, even in the steps where you think you know better. If it still won't help, let us know.

1. Prerequisites

1.1 Visual Studio 2013

Get **Visual Studio 2013 Express for Windows Desktop** or **Visual Studio 2013 Community Edition** (with Update 4) from

- <https://www.visualstudio.com/en-us/news/releasenotes/vs2013-community-vs>
- Go to this microsoft web page [Older Visual Studio](#) and scroll down to 2013.

If you use Windows XP or Vista, upgrade to Windows 7 or newer or try to use Visual Studio 2010 instead (you will need to build your own set of dependencies).

Install Visual Studio - you can disable all the optional features offered to save some space.

1.2 Git

Get the Git installation packages from <https://git-for-windows.github.io/>

- Install and let it register in the **PATH** environment variable
- The defaults for all the installation settings are fine except the following:
- On the **Adjusting your PATH environment**, select **Run Git from the Windows Command Prompt**.
- On the **Choosing CR/LF behavior** select **Checkout as-is, Commit Unix LF**.
- This is really important, the codebase uses Unix LF line endings and committing CR/LF makes the commits huge and hides their real content.
- If your git is already installed, please edit the **.git/config** file in your local working copy of the OpenCPN source tree and set **autocrlf = input** in the **[core]** section on top.
- If you want a tutorial, have a look at the series of articles starting at http://www.lostechies.com/blogs/jason_meridth/archive/2009/06/01/git-for-windows-developers-git-series-part-1.aspx
- References below to **Source Directory** are for where your local git repository is located + the project being worked on. Examples:

```
Source Directory Path: C:\\Users[username]\\Documents\\GitHub\\Ocpn
Build directory Path: C:\\Users[username]\\Documents\\GitHub\\Ocpn\\build
```

1.3 CMake

Get the latest CMake installation packages from <http://www.cmake.org>

Install and let it register in the **PATH** environment variable

1.4 POedit

Get the latest POedit installation package from <http://www.poedit.net>

- Install
- On 32bit Windows, add **C:\\Program Files\\Poedit\\GettextTools\\bin** to **PATH** environment variable. On a 64bit system, the path will be **C:\\Program Files (x86)\\Poedit\\GettextTools\\bin**

- On Windows 7 and later, open Computer, click System Properties, and in the left column click Advanced System Settings.
- On the Advanced tab, click on **Environment Variables** button and add the path in which **msgfmt.exe** resides to the PATH system variable.
- On Windows XP, right-click **My Computer**, select **Properties...**

1.5 NSIS

Current codebase

Get the current version 3.x of the NSIS installer from <http://nsis.sourceforge.net/> and install it.

1.6 wxWidgets 3

Get the latest 3.0 release from <http://wxwidgets.org/downloads/> (at the time of this writing 3.0.2)

Uncompress to your drive (we will assume to **C:\wxWidgets-3.0.2** in this text)

Note that git clone of opencpn contains 16 wxWidget files under "source\buildwin\wxWidgets, which are used for making installer packages.(NOT for compiling OpenCPN)

2 Compiling

2.0 Conceptual: Understanding how files are configured for Debugging and Release versions?

Example of a "Source" Directory under github:

```
C:\users[username]\Documents\GitHub\obeta**source**
source                                <-- Git clone or fetch from/into here
source\include                        <-- Git clone download of Include files
source\po                             <-- Git Internationalization files
source\src                            <-- Git clone download of Source Code
source\plugins                        <-- Git clone plugins from/into here.
source\data                           <-- source of some
source\buildwin                       <-- name_pi binaries & special dlls, exe put
here.
source\build                           <-- cmake commands are executed from this
directory.
source\build\plugins                   <-- copy name_pi.dll from the directory below
to here.
```

```
source\build\plugins\debug <-- copy compiled debug name_pi.dll from here
source\build\plugins\release <-- copy compiled release name_pi.dll from here
source\build\release <-- copy buildwin binaries & dlls to here for
running release.
source\build\release\plugins<--copy release name_pi.dll to here +
source\plugins\name_pi\data*.*.
```

The **source** might be named OpenCPN

2.1 Compile from the command line (recommended)

Note: This assumes that you have **Microsoft SDK 7.1A** installed on your pc. This was the last SDK that allowed building OpenCPN/wxWidgets for XP. If you do not have SDK 7.1A installed 'nmake' may run but the 'wxWidgets' that are built will not work with XP. You will get a 'not a valid Win32 application' message when trying to run OpenCPN in XP (OpenCPN built with these wxWidgets). So use wxWidgets built with SDK7.1A for compatibility with Windows XP.

Run the **Developer Command Prompt for VS2013**. Go to your wxWidgets build tree (**cd C:\wxWidgets-3.0.2\build\msw**) and build both **release** and **debug** configurations, compatible with Windows XP

```
RELEASE VERSION
nmake -f makefile.vc BUILD=release SHARED=1 CFLAGS=/D_USING_V120_SDK71_
CXXFLAGS=/D_USING_V120_SDK71_

DEBUG VERSION
nmake -f makefile.vc BUILD=debug SHARED=1 CFLAGS=/D_USING_V120_SDK71_
CXXFLAGS=/D_USING_V120_SDK71_
```

In case you are using Visual Studio 2010, the build commands are:

```
RELEASE
nmake -f makefile.vc BUILD=release SHARED=1

DEBUG
nmake -f makefile.vc BUILD=debug SHARED=1
```

2.2 Compile from Visual Studio IDE (optional)

This option is more work and not needed for 99% of people. Really, don't use it.

- In Visual Studio, open **wx_vc12.sln**
- Select **all projects** from the **Project Explorer** ,
- Right-click, select **Properties** ,

- Select **All Configurations** from the **Configuration:** dropdown on top
- and in **Configuration Properties** → **General** set the **Platform Toolset** to **Visual Studio 2013 - Windows XP (v120_xp)**
- Build both the **Debug** and **Release** DLL targets, **DO NOT** build the static libraries
- In case you are using Visual Studio 2010, use the **wx_vc10.sln** solution and don't change the platform toolset

2.3 Add wxWidgets to your PATH

In order for Cmake to find wxWidgets, you must add your wxWidgets root directory **C:[WXDIR]** (for example in place of WXDIR use 'C:\wxWidgets-3.0.2') to your **PATH environment variable**.

To be able to run debug builds and release builds *without install* add **C:[WXDIR]\lib\vc_dll** to your PATH.

After doing this, you have to restart the running programs (cmd.exe, cmake-gui, VisualStudio etc) to make sure they “see” the changed environment variables.

If you are unsure, restart Windows and everything will be set. If you don't do this you will have problems running your debug builds later.

If you have problems with cmake not finding your wxWidgets installation, try also creating another environment variable called **WXWIN** with a value of **C:[WXDIR]** (for example use 'C:\wxWidgets-3.0.2').

Also, try creating an environment variable called **wxWidgets_LIB_DIR=C:[WXDIR]\lib\vc_dll** and **wxWidgets_ROOT_DIR=C:[WXDIR]**. Again, don't forget to restart the running programs involved in the build.

3 Get the OpenCPN source

Run **Developer Command Prompt for VS2013** from Start menu → Programs → Microsoft Visual Studio → Visual Studio Tools

To get the source for the first time, from your local github directory or local git repository issue

```
git clone git://github.com/OpenCPN/OpenCPN.git
```

In case of error messages like this one:

```
"error: unable to create file  
buildwin/NSIS_Unicode/CopyNSISUnicodeRegKey.bat (Permission denied)"
```

observed under Windows 8.1, run the command from an Administrator console To update the code you cloned before, cd into the source directory cd OpenCPN and issue

```
git fetch --all
```

3.1 Get the binary dependencies

Note the Setup Batch File “Git_opencpn.bat” will download and expand OpenCPN_buildwin.7z into source\buildwin.

Get [OpenCPN_buildwin.7z](#) and extract the dependencies into your toplevel OpenCPN directory. The dependencies top directory is buildwin and hence the files and dirs will be placed under that directory.

(In case you are using Visual Studio 2010, you must build your own dependencies) and extract the archive into your toplevel OpenCPN source directory created by the clone operation above. The archive contains some binary files needed to link OpenCPN and produce the installer.

In case you need the PDB files for the prebuilt libraries (unlikely, really, if you don't know what for, you don't), get them from [here](#).

When extracting the libraries into the build tree, don't overwrite existing files. This will cause problems with pull requests. The **buildwin** directory after extraction should look like this:

Directory of C:\\Users[username]\\Documents\\GitHub\\Ocpn\\buildwin

Directories

crashrpt
expat-2.1.0
gtk
include
NSIS_Unicode
vc
wxWidgets

Files

archive.lib
archive.dll
liblzma.dll
lzma.lib
zlib1.dll
libcurl.dll
libcurl.lib
ocpn_gltest1.exe
Toolchain-mingw32.cmake

4 Build OpenCPN

- Run **Developer Command Prompt for VS2013**
 - From Start menu → Programs → Microsoft Visual Studio → Visual Studio Tools.
- Change Directory [CD] into your the topmost source directory.
- Create a directory named “build” under the topmost source directory.

```
mkdir build
```

4.1a - Configure "build" from VS Command Prompt:

- Use of the VS Command Prompt and CMake commands is recommended, rather than using the CMake-GUI interface.
- Change Directory [CD] into the “build” directory and then issue the cmake command which uses the large CMakeLists.txt located in the Source Directory (C:\Users[username]\Documents\GitHub\Ocpn) to determine to Operating System and then to set up the build directory for compilation and building commands to follow.

```
cd build
cmake -T v120_xp ..
```

In case you are using Visual Studio 2010, the command is **cmake ..**

The normal result:

```
C:\Users\Frederick\Documents\GitHub\Ocpn\build>cmake -T v120_xp ..
-- Building for: Visual Studio 12 2013
-- The C compiler identification is MSVC 18.0.31101.0
-- The CXX compiler identification is MSVC 18.0.31101.0
-- Check for working C compiler using: Visual Studio 12 2013
-- Check for working C compiler using: Visual Studio 12 2013 -- works
[removed a large section of output results here]
-- Generating done
-- Build files have been written to:
C:/Users/Frederick/Documents/GitHub/Ocpn/build
C:\Users\Frederick\Documents\GitHub\Ocpn\build>
```

NOTE: This is a good point to **Start over again** if you mess up copying the necessary files or are having troubles compiling & building. The first reasonable intermediate step is to remove all the files in the **build** directory and then start over by using **cmake -T v120_xp ..** from the **..\build** directory. That often will solve a configuration problem. Also you could just “git clone” another differently named OpenCPN repository and configure that again.

4.1b - Configure "build" Using Cmake-gui

(in case the previous way was too simple for you)

Run "**CMake (cmake-gui)**" from **Start menu → Programs → Cmake 3.2**. Fill in your source and build directories.

```
source = ...../OpenCPN
build = ...../OpenCPN/build
```

Click on the **Configure** button.

If you are asked to choose the generator, select "**Visual Studio 12**" (Or better say, select the version of VS you want to use for your build). The information which appeared is red and the Generate button stays disabled? Just hit **Configure** again...

Ignore GTK2_GTK_INCLUDE_DIR-NOTFOUND and wxwidgets_wxrc_EXECUTABLE_NOTFOUND.

- Click on the Generate button.
- Solution and project files should be created in your build directory.

IMPORTANT suggestion: Use **CMAKE GUI** tool to configure OpenCPN to verify that **wxWidgets_LIB_DIR** points to the **{root}/lib/vc_dll** directory. This check is necessary since the cmake FindWxWidgets module sometimes finds the wrong source and/or build config.

If you are using CMake version 3.0 or later you will get warnings about Policy CMP0043. These can be ignored.

4.2a - Compiling from the command line

- Run **Developer Command Prompt for VS2013** from Start menu → Programs → Microsoft Visual Studio → Visual Studio Tools.
- Change Directory [cd] into the **build** directory. Issue the command for a "release" build

```
cd build
cmake --build . --config release
```

The result output in the prompt window is lengthy because about 700mb of files are being created in the **build/release** directory from the cmake setup and the **src** directory C++ code files. The first time this command is run may end in ~50 warnings and possibly some errors, but the second or third time it is run the output should end with **0 Error(s)**. However **Warnings** are ok and should not affect operation.

```
Build succeeded.
    0 Warning(s)
    0 Error(s)
```


or issue the command for a “debug” build, from the Source Directory,

```
cd build
cmake --build . --config debug
```

The debug version is similar to compiling for release, but adds a number of files which track processes for the purpose of debugging. The first run will have ~50 warnings. It should also end with **0 Error(s)** after several re-runs of the command.

Note that if you don't use the `--config` parameter, a debug build is performed.

Wait for the builds to complete.

At this point the **build\debug** directory has 9 lib files, `opencpn.pdb`, `opencpn.ilc` and `opencpn.exe` and no subdirectories. The **build\release** directory does not contain `opencpn.ilc` or `opencpn.pdb`

```
C:\\Users[username]\\Documents\\GitHub\\Ocpn\\build\\debug & \\release
opencpn.exe
opencpn.exp
opencpn.ilc (debug directory only)
opencpn.pdb (debug directory only)
opencpn.lib
GARMINHOST.lib
MIPMAP.lib
NMEA0183.lib
S57ENC.lib
SYMBOLS.lib
TEXCMP.lib
WXCURL.lib
WXSVG.lib
```

Neither the **release** or **debug** will run properly at this point, because the necessary files and important DLLs are not available in several directories. Below we will show you several ways to copy the correct files to these directories.

4.2b - Compiling from Visual Studio

In Visual Studio, open the solution created by the **CMake command** earlier (Use the file **../build/OpenCPN.sln**). Compile the whole solution or individual projects. You must compile project **opencpn** before you can compile any plugins (to be fixed in the configuration process)

If you want to debug, don't forget to select **opencpn** as a start-up project. Once this is done, in the Solution Explorer (right panel) the project **opencpn** will be bold.

If you didn't add the WX DLL path to the PATH environment variable earlier, copy the needed WX DLLs to the build directory (Debug or Release, depending on which version you build). The DLLs can be found in **C:\${WXDIR}\\lib\\vc_dll** and you will need:

Debug: wxbase30ud_net_vc_custom.dll, wxbase30ud_vc_custom.dll, wxbase30ud_xml_vc_custom.dll, wxmsw30ud_adv_vc_custom.dll, wxmsw30ud_aui_vc_custom.dll, wxmsw30ud_core_vc_custom.dll, wxmsw30ud_gl_vc_custom.dll, wxmsw30ud_html_vc_custom.dll, wxmsw30ud_media_vc_custom.dll, wxmsw30ud_propgrid_vc_custom.dll, wxmsw30ud_qa_vc_custom.dll, wxmsw30ud_ribbon_vc_custom.dll, wxmsw30ud_richtext_vc_custom.dll, wxmsw30ud_stc_vc_custom.dll, wxmsw30ud_webview_vc_custom.dll, wxmsw30ud_xrc_vc_custom.dll

Release: wxbase30u_net_vc_custom.dll, wxbase30u_vc_custom.dll, wxbase30u_xml_vc_custom.dll, wxmsw30u_adv_vc_custom.dll, wxmsw30u_aui_vc_custom.dll, wxmsw30u_core_vc_custom.dll, wxmsw30u_gl_vc_custom.dll, wxmsw30u_html_vc_custom.dll, wxmsw30u_media_vc_custom.dll, wxmsw30u_propgrid_vc_custom.dll, wxmsw30u_qa_vc_custom.dll, wxmsw30u_ribbon_vc_custom.dll, wxmsw30u_richtext_vc_custom.dll, wxmsw30u_stc_vc_custom.dll, wxmsw30u_webview_vc_custom.dll, wxmsw30u_xrc_vc_custom.dll

See **6 - Running** below to prepare to run the Debug or Release build (from Visual Studio or otherwise) without “installing” - This involves using the compile setup plus copying certain essential and needed files to proper locations so the compile setup will run Opencpn.

5 Optional: Create the installer package

If you skipped the step 3.1 and did not do **3.3 Get the binary dependencies** yet, please go back there, otherwise you won't be able to create the package.

Build the **PACKAGE** project and **opencpn_[version]_setup.exe** is created in your build directory (replace X with the release and Y with the build number). Use the following command:

```
cmake --build . --target package --config release
```

OR after using

```
cmake --build . --config release
```

Then just type **cpack** in the MS VStudio command prompt to run the NSIS Install Packager.

This will create a new directory under the **build** directory called **_CPack_Packages**. You should find your install package under the **NSIS** sub-directory

```
C:\Users[username]\Documents\GitHub\Ocpn\build\_CPack_Packages\win32\NSIS
OR
source\build\_CPack_Packages\win32\NSIS
```

Now you can execute the NSIS Install Package file

```
..source\build\_CPack_Packages\win32\NSISopencpn_[version]_setup.exe
```

to install a recently compiled working version of OpenCPN.

You will also find a subdirectory `..\NSIS\opencpn_[version]_setup` with all the files used to create this install package.

- This directory is very useful to determine if a file has been included or not.
- If not, and a file is needed for the installation you can manually add it to the `\buildwin` directory and the file will be copied into the `\NSIS\opencpn_[version]_setup` directory [a useful trick sometimes].
- The “CMake -target package” or “cpack” command creates the NSIS Install directory the NSIS file `opencpn_[version]_setup.exe`. This executable contains all the files in the directory in compressed format.
- A listing of the files and folder in `NSIS\opencpn_[version]_setup` (dated: 2/18/2017). The [NSIS Installation Directory](#) installation directory used to create the Install package.

If you do not intend to use the “release” NSIS Package Installation “exe” to create your “Release” OpenCPN version, you will need to manually complete the actions shown in this [Table for Creation of Operational "Release" Version](#) .

The simple NSIS system command `cpack` or adding `-target package` to cmake commands, takes care of all this for you, so use it when you can! It is good for “packaging” Windows installation of Opencpn.

Currently the installer packs the DLLs from the git repository into the package. You have to replace them with your custom built DLLs after the installation if you want to experiment with different versions and build settings of the wxWidgets libraries. This is a side matter to the focus of compiling OpenCPN, which is significant when developing a plugin or working on code.

6 Setup Build, Release & Debug Folders

Both Debugging and running Release (bypassing installation) require certain files to be copied.

1. Debug Build requires that certain files be copied to certain directories.
2. Running OpenCpn in portable mode from the source\build\release directory (bypassing the installation process) requires that certain files be copied to the source\build\release directory. Note: If you are using NSIS Package Installation and do not plan on running Opencpn from source\build\release, copying the files to source\build\release is not necessary.

There are two choices for setting up & maintaining the build, release and debug directories:

1. [6.1 Setup Batch Files](#) below (may be the quickest and easiest method).
2. [Step by Step Manual copy of files](#) as shown below.

6.1 Setup Batch Files

NOTE: Batch files are “Beta”

Maintenance and Updates

When you have git fetched & pulled or changed files, these batch files can be Re-run: config.bat, build.bat, dbbuild.bat, dbcopy.bat as needed before using MS VStudio, Using the NSIS Batch or Running the release version directly without installation.

1. [git_opencpn.bat.doc](#) NOTE: only used for a clean installation from [\github](#)
2. [config.bat.doc](#) del cmakeCache.txt, Copy files.
3. [dbbuild.bat.doc](#) Run cmake -build . -config debug & call dbcopy.bat
4. [dbcopy.bat.doc](#) Copy necessary dll files.
5. [build.bat.doc](#) *Run cmake -build . -config release -target package
6. [clean.bat.doc](#) Run MS VStudio "clean"

Remove the ".doc" from these batch files before running from the command prompt.

CopyFiles.bat.doc Håkan's single batch file - Another alternative [Copyfiles-Hakan.bat.doc](#) Hakan has also provided a useful single batch file which is placed in the source\build directory and executed using the command prompt to copy all the needed files to the various directories. This batch file does not execute any cmake commands (unlike the first Git_opencpn.bat System with 6 batch files) NOTE: This batch file has not been tested as completely, so it is really BETA at this point!!

4 Internal Plugins Note about copy of Plugin Release and Debug versions *_pi.dll and *_pi\data folders Both Batch File systems copy the needed files for the **4 internal plugins**, following the plugins compilation structure. This includes the data directories and proper files for the 4 internal plugins, for Debug to build\plugins and for Release to build\release\plugins. Please note that the debug folder build\plugins will only run "debug" plugins otherwise there will be an error.

Plugin Data Directories

	Source	Destination
Debug	Copy GitHub\obeta\source\plugins*_pi\data*.*	source\build\plugins*_pi\data
Release	Copy GitHub\obeta\source\plugins*_pi\data*.*	source\build\plugins*_pi\data

Plugin *_pi.DLL Files (note source is different for debug & release)

	Source	Destination
Debug	Copy source\build\plugins\chartdldr_pi\debug*_pi.dll	source\build\plugins*_pi\data
Release	Copy source\build\plugins\chartdldr_pi\release*_pi.dll	source\build\release\plugins*_pi\data

6.1a Setup Batch Files

NOTE: These Batch files are "Beta" Still Testing & Checking

Use these batch files for a quicker way to complete Sections 3 through 7. Upon completion use MS VStudio Debug or run OpenCPN /p directly from the source\build\release directory.

Git_opencpn.bat is the first batch file used, it will "git clone opencpn", get and install the binary

dependencies, provided you have Powershell (most Windows OS have it) and 7z installed, execute "cmake -T v120_xp .." and then call the other batch files to complete the setup.

Git_opencpn.bat will simplify installation once you have completed:

1. Prerequisites - Visual Studio 2013, CMake, Poedit, wxWidgets 3.0.2, Github working.
2. Compile to just before **3. Get the OpenCPN source**

Developed by TransmitterDan and Håkan, modified by rgleason & tested. Read the instructions carefully and use them at your own risk. To start:

- Download and remove the ".doc"
- Copy the files to **C:\Users\[username]\Documents\GitHub**
- Make sure you have 7z and Powershell installed.
- From the VS Command prompt in **\github**, execute **git_opencpn**.

What do the Setup Batch Files Do?

1. **Git_opencpn.bat** batch file will
 1. Make a directory obeta, change directory to obeta
 2. Execute git clone <https://github.com/OpenCPN/OpenCPN.git>
 3. making directory OpenCPN and downloading github source files.
 4. Change directory to OpenCPN
 5. Download binary files with Powershell and expand them with 7z
 6. Make directory Build and change directory to Build
 7. Copy the batch files to OpenCPN\build where they will be executed.
 8. Call Config.bat, then dbbuild.bat, then build.bat
2. Config.bat
 1. del CMakeCache.txt
 2. cmake -T v120_xp ..
 3. Copy files to source\build, source\build\plugins, source\build\release, source buildwin
3. dbbuild.bat
 1. cmake -build . -config debug
 2. cmake -build . -config debug
 3. call dbcopy.bat
4. dbcopy.bat
 1. Copies necessary dll files to source\build\debug
 2. Copies 4 internal debug plugin dll to build\plugins
 3. Copies 4 internal release plugin dll to build\release\plugins
5. build.bat
 1. del opencpn*.exe
 2. cmake -build . -config release
 3. cmake -build . -config release -target package

The batch files will complete the setup from Step 3 to 7. Then Start Debugging in MS VStudio and find an NSIS Installation package in source\build provided the prerequisite for [1.5 NSIS](#) is completed. This system has 6 batch files executed from the command prompt. These files are being testing now, still

beta.

Download the Setup Batch Files

1. [git_opencpn.bat.doc](#) Mkdir, binary files, copy files, call config, dbbuild & build.bat
2. [config.bat.doc](#) del cmakeCache.txt, cmake -T v120_xp .., Copy files.
3. [dbbuild.bat.doc](#) Run cmake -build . -config debug & call dbcopy.bat
4. [dbcopy.bat.doc](#) Copy necessary dll files.
5. [build.bat.doc](#) Run cmake -build . -config release -target package
6. [clean.bat.doc](#) Run MS VStudio "clean"

Once executed you should then be able to use MS VStudio to debug Opencpn After the full installation is completed, re-run config.bat, build.bat, dbbuild.bat as needed.

CopyFiles.bat.doc Håkan's single batch file - Another alternative [Copyfiles-Hakan.bat.doc](#) Remove the ".doc" from these batch files before running from the command prompt.

Hakan has also provided a useful single batch file which is placed in the source\build directory and executed using the command prompt to copy all the needed files to the various directories. This batch file does not execute any cmake commands (unlike the first Git_opencpn.bat System with 6 batch files)

NOTE: This batch file has not been tested as completely, so it is really BETA at this point!!

6.2 Step by Step Manual Copy of Setup Files

Complete the following.

=== Create two new "uidata" folders ===

```
Files from source/src/bitmaps to uidata
```

```
-----
```

```
styles.xml
toolicons_traditional.png
toolicons_journeyman.png
toolicons_journeyman_flat.png
iconAll.png
iconMinimum.png
iconRMinus.png
iconRPlus.png
iconStandard.png
```

```
3 SVG Directories and files from source/data/svg to uidata
```

```
-----
```

```
journeyman
journeyman_flat
```

traditional

Styles (custom style files from source/styles to uidata)

Copy the individual style.xml files into uidata

Copy the files and folders listed above

- For Debug build into the **source\build\uidata** directory (do not use **source\build\Debug**)
- For Release build into the **source\build\Release\uidata** directory.

Shortcut: If you have run the NSIS Install Package “cpack”, just copy
..\build_CPack_Packages\win32\NSIS\opencpn_[version]_setup\uidata to the folder.

Copy Six Data Directories

Six directories from source\data

doc
sounds
tcdata
wvsdata
gshhs
s57data

Copy the six Data folders above from **source\data**

- For Debug build into the **source\build** directory (do not use **build/Debug**)
- For Release build into the **source\build\Release** directory.

Copy Necessary Individual Files and DLL files

11 Dll files and Individual files from various locations

Individual files	From
zlib1.dll	source\buildwin\zlib1.dll
libcurl.dll	source\buildwin\libcurl.dll
ocpn_gltest1.exe	source\buildwin\ocpn_gltest1.exe
libpng16.dll	source\buildwin\gtk\libpng16.dll
cairo.dll	source\buildwin\gtk\cairo.dll
fontconfig.dll	source\buildwin\gtk\fontconfig.dll
iconv.dll	source\buildwin\gtk\iconv.dll
libxml2.dll	source\buildwin\gtk\libxml2.dll
pixman-1.dll	source\buildwin\gtk\pixman-1.dll

expat.dll	source\buildwin\expat-2.1.0\expat.dll
msvcpl120.dll	source\buildwin\vc\msvcpl120.dll
msvcr120.dll	source\buildwin\vc\msvcr120.dll

license.txt	source\data\license.txt
-------------	-------------------------

16 wxWidget Files

Files	From
All Files	source\buildwin\wxWidgets*.*
wxbase30u_net_vc_custom.dll	(only for release)
wxbase30u_vc_custom.dll	(only for release)
wxbase30u_xml_vc_custom.dll	(only for release)
wxmsw30u_adv_vc_custom.dll	(only for release)
wxmsw30u_aui_vc_custom.dll	(only for release)
wxmsw30u_core_vc_custom.dll	(only for release)
wxmsw30u_gl_vc_custom.dll	(only for release)
wxmsw30u_html_vc_custom.dll	(only for release)
wxmsw30u_media_vc_custom.dll	(only for release)
wxmsw30u_propgrid_vc_custom.dll	(only for release)
wxmsw30u_qa_vc_custom.dll	(only for release)
wxmsw30u_ribbon_vc_custom.dll	(only for release)
wxmsw30u_richtext_vc_custom.dll	(only for release)
wxmsw30u_stc_vc_custom.dll	(only for release)
wxmsw30u_webview_vc_custom.dll	(only for release)
wxmsw30u_xrc_vc_custom.dll	(only for release)

Crash Report

Files	From
PrivacyPolicy.txt	source\buildwin\crashrpt\PrivacyPolicy.txt
CrashRpt1403.dll	source\buildwin\crashrpt\CrashRpt1403.dll (only for release)
crashrpt_lang.ini	source\buildwin\crashrpt\crashrpt_lang.ini (only for release)
CrashSender1403.exe	source\buildwin\crashrpt\CrashSender1403.exe (only for release)
dbghelp.dll	source\buildwin\crashrpt\dbghelp.dll (only for release)

Copy the files (as noted above):

- For Debug build into the **source\build\Debug** directory
- For Release build into the **source\build\Release** directory.

7 Running OpenCPN from "build" directory

Thus bypassing installation.

Running for the first time

Have you copied the necessary files as listed in 6.2 OR used the Batch Files in 6.1? Keep in mind what you're intent is, in running for the first time!

1. For Debug Build use **MS VStudio** and open **source\build\opencpn.sln**
2. For Release Build execute portable **opencpn /p** from **source\build\release**
3. For Release run an NSIS Installation Package "cpack" and install the exe.

To run the [first time] issue the following command from a command prompt in the **build/Debug** or **build/Release** directory:

```
opencpn.exe /p
```

This will generate an opencpn.ini file in the current directory as well as create the opencpn.log file.

Note: **opencpn.exe /p** portable switch must be used **every time** otherwise when **opencpn** is used and we go to options > plugins there is an image format error, and the programdata\opencpn\opencpn.ini file is used.

8 Something does not work as expected?

Before getting desperate, **read your opencpn.log logfile**, it is likely that the problem is clearly identified there.

From:
<https://opencpn.org/wiki/dokuwiki/> - OpenCPN Manuals

Permanent link:
https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:developer_manual:developer_guide:compiling_windows:compile_win_ov4_wx3.0.2

Last update: 2021/03/25 15:33

